File I/O, command-line arguments and exceptions

A secret message has been embedded into a couple of PNG images by means of steganography [1]. Unfortunately other images, containing no hidden information, have been mixed into our images of interest. At least no images have been lost, but there might be some corrupted ones.

Problem

You received a ZIP file, containing PNG images. Write a Java program where the path to this ZIP file is given via the command-line. The program shall first open the ZIP file and extract all its entries. Then it has to check for hidden messages in each of its entries. A library, *magic.jar*, is provided which is able to check for hidden messages within a given PNG image (given by the path to the file).

After processing all the images, your program shall output the extracted messages. Note that the words of the message might not be in order, i.e. the images might not be processed in correct grammatical order of the hidden message.

Command-line interface

When calling the Java program, one command-line argument has to be given, the path to the ZIP file. For information of Java command-line arguments see [2].

Hidden message searcher

Next you need to implement a class, *StegoSearcher*, which walks through the given ZIP file and calls the *magic.jar* library to extract hidden data. See [3] on how to handle ZIP files in Java. To get a path to unpack the ZIP file contents (the PNG images), use *java.io.Files* to create a temporary directory. Make sure to **delete unpacked ZIP file contents** at the end of your program!

The magic.jar library provides the following method:

```
package stego;

String StegoImageExtractor.extract(String pngPath)
      throws InvalidPNGException;
```

Use this method to extract messages from the PNG images within the ZIP file. The return value of this method is the secret message extracted from the given image. Be careful to handle corrupted PNG images correctly, i.e. take care of the exception which might be thrown by this method.

Grading

Minimal implementation (70 points)

Implementing the handling of command-line arguments and searching through one given ZIP file (easy.zip), without generating errors, is awarded with up to 70 points. The final grading also considers your explanation of the source code.

All ZIP files and custom exception (15 points)

Besides handling the exception from the *magic.jar* library your *StegoSearcher* class shall also throw a custom *InvalidZIPException* in case some problems occurred while reading the ZIP file and its contents. Make sure to handle all exceptions in your test program (i.e. within main) correctly.

Console-based progress bar (15 points)

While the contents of the given ZIP file are analyzed, the console shall display a progress bar (e.g. put together using '#' signs) to indicate the amount of time until the hidden message search is done. One way to statistically compute the amount of the left is to take the total number of bytes of all file within the ZIP file and contrast it to the number of bytes already processed.

Notes

A quick overview on how to import the magic.jar library into your project can be found at https://www.youtube.com/watch?v=UtzAf8tyuAM.

References

[1] Steganography: Hiding Data Within Data, Gary C. Kessler, September 2001, http://www.garykessler.net/library/steganography.html.
[2] Command-Line Arguments, Oracle Java Documentation, 2014, http://docs.oracle.com/javase/tutorial/essential/environment/cmdLineArgs.html.
[3] How To Decompress Files From A ZIP File, mykong, January 2010, http://www.mkyong.com/java/how-to-decompress-files-from-a-zip-file/.